AF
D

PATENT

I hereby certify that on the date specified below, this correspondence is being deposited with the United States Postal Service as first-class mail in an envelope addressed to MAIL STOP APPEAL BRIEF-PATENTS, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

11-19-07
Date

Joanne Bourguignon
Joanne Bourguignon

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants	:	Eric P. Gibbs et al.
Application No.	:	09/975,749
Filed	:	October 10, 2001
For	:	SYSTEM AND METHOD FOR DATA TRANSFER OPTIMIZATION IN A PORTABLE AUDIO DEVICE

Examiner	:	Daniel R. Sellers
Art Unit	:	2644
Docket No.	:	35073.002
Date	:	November 19, 2007

Attention: Board of Patent Appeals and Interferences
Commissioner for Patents,
P.O. Box 1450,
Alexandria, VA 22313-1450

TRANSMITTAL OF REPLY BRIEF

Sir:

Transmitted herewith in triplicate is the Reply Brief with respect to the Examiner's Answer mailed on March 18, 2005. This Reply Brief is being filed pursuant to 37 CFR § 41.41(a)(1) within two months of the date of the Examiner's Answer.

We believe that no fee is required for the filing of this Reply Brief. But, at anytime during the pendency of this application, please charge any fees required or credit any overpayment to Deposit Account No. 50-2976 pursuant to 37 CFR 1.25. Additionally, please charge any fees to Deposit Account No. 50-2976 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. This notice is being submitted in duplicate.

Respectfully submitted,
Eric P. Gibbs et al.
OLYMPIC PATENT WORKS PLLC

Robert W. Bergstrom
Robert W. Bergstrom
Registration No. 39,906

Enclosure:
Copy of Transmittal

Olympic Patent Works PLLC
P.O. Box 4277
Seattle, WA 98194-0277
206.621.1933 telephone
206.621.5302 fax

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:

Applicants	:	Eric P. Gibbs et al.
Application No.	:	09/975,749
Filed	:	October 10, 2001
For	:	SYSTEM AND METHOD FOR DATA TRANSFER OPTIMIZATION IN A PORTABLE AUDIO DEVICE

Examiner	:	Daniel R. Sellers
Art Unit	:	2644
Docket No.	:	35073.002
Date	:	November 19, 2007

REPLY BRIEF UNDER 37 CFR 41.41(a)(1)

Mail Stop: Appeal Briefs – Patents
Commissioner of Patents and Trademarks
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In response to the Examiner's Answer dated September 19, 2007, applicant's reply as follows: .

REAL PARTY IN INTEREST

The real party in interest is Mark E. Phillips having an address of 720 – 3rd Ave., Suite 1100, Seattle, Washington 98104.

RELATED APPEALS AND INTERFERENCES

Applicant's representative has not identified, and does not know of, any other appeals of interferences which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

STATUS OF AMENDMENTS

No Amendment After Final is enclosed with this brief. The last Amendment was filed September 26, 2006.

SUMMARY OF CLAIMED SUBJECT MATTER

Independent Claim 28

Claim 28 is directed to a portable media player (Page 3, lines 15-28; 100 in Figure 1) that includes a processor (102), a random-access-memory component (104), a codec component (114), a non-volatile, mass-storage component (126), and a battery power supply (132). The processor executes commands. The random-access-memory component stores compressed data in more than two different random-access-memory buffer areas (Page 12, lines 4-14), each random-access-memory buffer lockable and unlockable by the processor (Page 17, line 26 - page 18, line 7). The codec component, controlled by the processor, reads compressed data from a locked random-access-memory buffer. The locked random-access-memory buffer is selected from among the more than two different random-access-memory buffer areas and is locked by the processor to prevent writing of the locked random-access-memory buffer by another component. The codec component generates a decompressed signal from the compressed data (Page 4, lines 26-29) that is, in turn, rendered by a data-rendering component (Page 5, lines 19-27). The non-volatile mass-storage component stores compressed data and that writes compressed data, under control of the processor, to unlocked random-access-memory buffers (Page 6, lines 5-16). The battery power supply provides electrical power to the processor, random-access memory component, codec component, data-rendering component, and non-volatile, mass-storage component (page 6, line 24 - page 7, line 2).

Dependent Claims 29-37

Claim 29 is directed to the portable media player (Page 3, lines 15-28; 100 in Figure 1) of claim 28 wherein the processor (102) continuously monitors progress of the codec component (114) in decompressing data in order to power up the non-volatile, mass-storage component (126), direct the non-volatile, mass-storage component to write additional compressed data to multiple random-access-memory buffers (104) and redirect the codec component to read the additional compressed data from the multiple random-access-memory buffers so that the codec component can continue to generate a decompressed signal without interruption, and power-down the non-volatile, mass-storage component (Page 9, line 29 - page 10, line 6). Claim 30 is directed to the portable media player (Page 3, lines 15-28; 100 in Figure 1) of claim 29 wherein the processor, following reception of a fast-forward command that redirects rendering, by the data-rendering component, of compressed data starting at a desired location within a compressed-data sequence not currently stored within the more than two different random-access-memory buffer areas, directs the non-volatile, mass-storage component to write compressed data, starting at a location prior to the desired location in the compressed-data stream and ending at a location following the desired location in the compressed-data stream, to multiple random-access-memory buffers (Page 15, lines 9-25). Claim 31 is directed to the portable media player (Page 3, lines 15-28; 100 in Figure 1) of claim 29 wherein the processor, following reception of a rewind command that redirects rendering, by the data-rendering component, of compressed data starting at a desired location within a compressed-data sequence not currently stored within the more than two different random-access-memory buffer areas, directs the non-volatile, mass-storage component to write compressed data, starting at a location prior to the desired location in the compressed-data stream and ending at a location following the desired location in the compressed-data stream, to multiple random-access-memory buffers (e.g., page 15, line 26 - page 16, line 11). Claim 32 is directed to the portable media player (Page 3, lines 15-28; 100 in Figure 1) of claim 29 wherein the processor, following reception of a rewind command that redirects rendering, by the data-rendering component, of compressed data starting at a desired location within a compressed-data sequence not currently stored within the more than two different random-access-memory buffer areas, directs the non-volatile, mass-storage component to write compressed data, starting at a location prior to the desired location in the compressed-data stream and ending at a location at which subsequent compressed-data of the compressed-data sequence is already stored in the more than two different random-access-memory buffer areas, to multiple random-access-memory buffers (Page 14, lines 4-24). Claim 33 is directed to the portable media player (Page 3, lines 15-28; 100 in Figure 1) of claim 29 wherein the processor, following reception of a fast-forward command, predicts portions of a compressed-data sequence that are likely to be accessed by additional fast-forward commands and directs the non-volatile, mass-storage component to write predicted portions of the compressed data to multiple random-access-memory buffers (Page 16, lines 12-21). Claim 34 is directed to the portable media player (Page 3, lines 15-28; 100 in Figure 1) of claim 29 wherein the processor minimizes the number of times that the processor powers up the non-volatile, mass-storage component (Page 3, lines 2-14). Claim 35 is directed to the portable media player (Page 3, lines 15-28; 100 in Figure 1) of claim 29 wherein the processor minimizes the duration of time during which the non-volatile, mass-storage component is powered up (Page 3, lines 2-14). Claim 36 is directed to the portable media player (Page 3, lines 15-28; 100 in Figure 1) of claim 29 wherein the processor locks only a single random-access-memory buffer at any point in time (Page 17, line 26 - page 18, line 27). Claim 37 is directed to the portable media player (Page 3, lines 15-28; 100 in Figure 1) of claim 29 wherein the compressed data is a compressed audio signal and the decompressed signal is a decompressed audio signal.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. The rejection of claims 28-37 under 35 U.S.C. §103(a) as being unpatentable over Birrell et al., U.S. Patent No. 6,332,175 B1 in view of Biliris et al., U.S. Patent No. 5,720,037.

ARGUMENT

Claims 28-37 are pending in the current application. In an Office Action dated February 13, 2007, the Examiner rejected claims 28-37 under 35 U.S.C. §103(a) as being unpatentable over Birrell et al., U.S. Patent No. 6,332,175 B1 ("Birrell") in view of Biliris et al., U.S. Patent No. 5,720,037 ("Biliris"). Applicants respectfully traverse the 35 U.S.C. §103(a) rejections of claims 28-37.

ISSUE 1

1. The rejection of claims 28-37 under 35 U.S.C. §103(a) as being unpatentable over Birrell et al., U.S. Patent No. 6,332,175 B1 in view of Biliris et al., U.S. Patent No. 5,720,037.

The purpose of the present Reply Brief is to briefly respond to the Examiner's

Answer, dated September 19, 2007. Appellants rely on the original arguments filed in the Appeal Brief on May 14, 2007, and respond only to the Examiner's newly stated arguments beginning in section 10 of the Examiner's Answer.

In particular, Appellants wish to respond to section 10 of the Examiner's Answer, in which the Examiner states:

> Regarding claim 28, the combination of Birrell and Biliris teaches these features. Specifically, the claimed limitation "each random-access-memory (RAM) buffer lockable and unlockable by the processor" is inherent in the teaching of Birrell. The Office reads claim language using the broadest reasonable interpretation consistent with the specification. The specification discloses the applicant's definition of locked and unlocked, wherein locked is described as available for read operations and not write operations and unlocked is described as available for read operations but not write operations and unlocked is described as available for both operations (p. 17, line 27 - p. 18, line 8). Birrell inherently teaches a buffer that is locked for read only operations, because the rewind buffer created by Birrell must have some indication to retain and not overwrite data in RAM (Col. 6, line 64 - Col. 7, line 5). If Birrell did not lock portions of the RAM, which contain play data, then play control logic would overwrite all of the data stored in RAM. Therefore the teachings of retaining previously-played data in RAM reads on a lockable RAM buffer. Furthermore, this also makes inherent an unlockable RAM buffer, because at any instant in time when the current play position in the buffer advances, a new portion of previously-played data, which was previously protected in the RAM by a retention threshold, will become available to overwrite. Birrell's invention would not function properly if it could not free memory in the RAM after it is retained for some portion of time, because it would run out of free memory to store more audio samples for future playback.
>
> Methods of coordinating access to memory and locking, by use of semaphores and spin locks are discussed by on pages 7 to 8 of applicant's arguments. The examiner does not understand how this definition of locking is to be read into the claims. The claim language does not invoke 35 USC 112 sixth paragraph limitations, and as discussed above the term "locking" is given a reasonably broad interpretation with respect to the specification. The applicant's specification, starting on page 17, line 26, recites:
>
> > Part of the efficiency provided by the buffering techniques of the system 100 is that only one buffer is "Locked" for reading to the CODEC 114 while the other buffers are available for read/write operations.
>
> The broadest reasonable meaning appears to be where locked is defined as allowing read-only operations, and unlocked is defined as allowing both read and write operations.
>
> With respect to section 11, the Examiner still asserts that Birrell teaches "buffers that are needed for rewind are locked and only accessible by read commands." Birrell discloses these teachings from column 6, line 64 to column 7, line 5. Birrell clearly states, "the final portion of the previously-

> played data will be **retained** in case the user wishes to reverse the direction of play." (emphasis added) This teaches that some portion of data in memory is kept, or locked, because the idea of retention does not work if the data can be overwritten. Only when the previously-played data falls outside this final portion, will the buffer be freed, or unlocked, so that new data can be written to it.

Appellant's representative confesses to being astonished at this statement, for many different reasons. It is both incorrect and based on, in part, classic flaws in reasoning and logic. Most disturbingly, Appellant's representative is left with the impression that the Examiner does not understand basic computer science, or the subject matter of the cited references and the current application.

The Examiner asserts: "The specification discloses the applicant's definition of locked and unlocked, wherein locked is described as available for read operations and not write operations and unlocked is described as available for both operations (p. 17, line 27 - p. 18, line 8)." The cited portion of the specification is provided below:

> Part of the efficiency provided by the buffering techniques of the system 100 is that only one buffer is "Locked" for a reading to the CODEC 114 while the other buffers are available for read/write operations. In the examples illustrated in Figures 4-9, sixteen buffers are allocated as part of the buffer 124. Thus, only 1/16 of the total buffer space is locked for data transfer to the CODEC 114 and is thus unavailable for other read/write operations. However, the remaining 15/16 of the total buffer space are available to be filled each time the storage device 126 is activated. Such operation is in sharp contrast to a typical buffering operation in which a buffer is allocated into two portions with only one-half of the buffer space available for read/write operations while the other half of the buffer space is locked for data transfer operations to the CODEC.

Appellant's representative cannot find a definition of "locked" and "unlocked" in this passage. In fact, the specification was written for an audience with understanding of basic computer science, and the author of the specification assumed that anyone with an understanding of basic computer science would well understand the term "locked." This passage states that only one of 16 buffers is locked for use by a CODEC, and therefore unavailable for other read/write operations. The term "locked" has a very well understood meaning in computer science. It means, in general, that one or a set of processes can access a locked entity or objects, while others cannot. The term "locked" as used in computer science is quite similar to the term "locked" in common English. One, for example, locks a door in order to prevent others from opening the door. Those with keys to the door can unlock it and pass through the door, while those without keys cannot. Furthermore, the Examiner's assumed definitions, "wherein locked is described as available for read operations and not write operations and unlocked is described as available for both operations," is incorrect and does not in any way follow from the above-quoted passage of the specification. The one buffer that is locked is available for read access by the CODEC, but unavailable for accesses of any kind by other entities, specifically a system processor or disk-drive controller. The above-quoted statement is analogous to stating, "The house is locked so that only a person with a key can get in, while others cannot." Such a statement does not define the word "locked," but instead describes the consequences of the fact that something is locked. The meaning of the term "locked" is assumed to be understood by the reader of such a sentence, just as the meaning of the term "locked," when used in a computer-system context, is assumed to be understood by those familiar with computer systems by the author of the above-quoted passage.

The Examiner states: "Birrell inherently teaches a buffer that is locked for read only operations, because the rewind buffer created by Birrell must have some indication to retain and not overwrite data in RAM (Col. 6, line 64 - Col. 7, line 5)." The quoted portion of Birrell is provided below:

> In addition, in one embodiment, play control logic will not completely overwrite the data in RAM with data from disk 104 once the threshold is reached. Instead, the final portion of the previously-played data will be retained in case the user wishes to reverse the direction of play. Thus, in this embodiment, the amount of data comprising the final portion would be at least as great as the rewind speed multiplied by the amount of time it takes to access disk 104 and copy data from disk 104 to RAM 108.

There is not a single mention or suggestion of locking anywhere in this statement. The Examiner's statement makes absolutely no sense. The *rewind buffer* does not need to have some indication to retain and not overwrite data in RAM. All that is needed is what is disclosed by Birrell, namely that the play control logic be able to determine how far the play control logic has written the RAM with respect to a final portion of the previously-played data so the play control logic does not overwrite the previously-played data. Similarly, were one overwriting data in a 10-byte linear array, and wanted to preserve the last byte in the linear array, one could preserve that last byte by writing elements of the linear array up through the byte preceding the last byte. No locking would be necessary. The same is true even were two cooperating processes sharing the linear array, and had both processes agreed ahead of time not to overwrite the last byte. Locking is only needed when different threads and/or processes access a common object without being able to easily adhere to a convention for access that would prevent the different threads and/or processes with interfering with one another, as, for example, by altering data by one process needed in its original form by another. In operating systems that support concurrently executing processes and threads written by application programmers, for example, locking is a valuable and needed technique for preventing processes and threads from interfering with one another. In many cases, locking is less expensive than trying to prevent interference by managing complex alternative protocols and data otherwise needed to manage access to shared objects. However, when concurrent access can be easily managed without locking, only an extremely naïve or stupid programmer would employ locking, because, as discussed in the Appeal Brief, locking is quite computationally expensive, and can lead to greatly increased latencies and the additional risk of deadlock. As with most things in computer science and engineering, there are many different ways to accomplish sharing of objects by concurrently executing threads or processes. Locking is but one approach, useful in certain cases, and quite inefficient in others. To suggest that locking is inherent because a process needs to not overwrite a known portion of a buffer is nothing short of absurd. The cited portion of Birrell does not mention, suggest, or even remotely imply any kind of locking. The statement is analogous to concluding that, because a house is to be painted, the house will necessarily be red following the painting. There are many possible colors, and, absent additional knowledge, there is no reason to conclude that the particular color chosen for the house will be red.

The Examiner's next assertion is an example of drawing conclusions from an incorrect statement: "Furthermore, this also makes inherent an unlockable RAM buffer, because at any instant in time when the current play position in the buffer advances, a new portion of previously-played data, which was previously protected in the RAM by a retention threshold, will become available to overwrite." No lock was suggested or implied, so there is absolutely no reason to suspect that something not protected by a lock will subsequently need to be unlocked. A retention threshold is not a lock. A threshold is a numerical value. Moreover, were the Examiner to read the cited reference, particularly the paragraph beginning on line 43 of column 6, the Examiner would realize that, in the described implementation, the RAM is overwritten all the way up to the threshold, and perhaps well beyond the threshold. The retention threshold is concerned with ensuring that there is enough data in RAM to allow continuous play even when the disk needs to be restarted. It has nothing to do with preventing overwrite of the final portion of the data in RAM or, in other words, protecting the final portion of the previously-played data.

The Examiner observes that, "Birrell's invention would not function properly if it could not free memory in the RAM after it is retained for some portion of time, because it would run out of free memory to store more audio samples for future playback." True enough, but this has absolutely nothing at all to do with locking. As one example, it is quite common to use circular buffers to buffer input characters received from a keyboard for consumption by input-character-processing routine in an operating system. When one can ensure, through interrupt priorities and knowledge of the system, that the input-character-processing routine will run with sufficient frequency to process a certain number of characters per second, then, by using a circular buffer of sufficient size, the operating system can ensure that the circular buffer will not be overwritten by keyboard input before the input-character-processing routine can consume the characters. Many first-year undergraduate students of computer science are required to write such routines in assembly language. No locking of any kind is required. Use of locking would have slowed input processing to a crawl in many older computer systems.

The Examiner states: "Methods of coordinating access to memory and locking, by use of semaphores and spin locks are discussed by on pages 7 to 8 of applicant's arguments. The examiner does not understand how this definition of locking is to be read into the claims." The language was not intended to be read into the claim. The discussion was meant to refresh the Examiner's recollection of what is meant by "locking" in computer science, and how locking is generally implemented. The language is not a definition of locking. The terms "locking," "lock," and "lock" are extremely well understood in computer science and computer engineering, These terms do not need to be defined to even an undergraduate student of computer science.

The Examiner states:

> Birrell clearly states, "the final portion of the previously-played data will be **retained** in case the user wishes to reverse the direction of play." (emphasis added) This teaches that some portion of data in memory is kept, or locked, because the idea of retention does not work if the data can be overwritten.

Again, this is an absolutely incorrect and illogical statement. For example, one could easily write a program in which a first thread is launched to write the contents of an array, and them that first thread and multiple subsequently launched threads read the data from the array.

Absolutely no locking is required to preserve the data in the array. In addition, one thread could overwrite the data in the array, up to some threshold array element, as often as it wished, and the remaining elements of the array would be preserved - without any need for locking. In a real time system, two processes can be easily implemented, without locking, so that one process continuously writes data to a circular memory buffer for another process to continuously read, simply by arranging for the writing process to write well ahead of the where the reading process is reading, and using a high-water mark to indicate the current last-written word in memory and a high-water mark to indicate the current last-read word in the memory. In these cases, both processes would have the ability to read and write anywhere in memory - but - by being written to avoid conflicts with one another, there would not be a need for locking. Locking is only needed when routines cannot be easily written to avoid conflicts. Here again, the Examiner begins with a flawed concept and lack of understanding of terms to draw unwarranted conclusions. The terms "retained," "kept," and "locked" are not equivalent, and have almost nothing to do with one another. To retain or keep data means to preserve the data in a form in which it can be subsequently read, or accessed. To lock data means to prevent certain processes and threads from accessing the data. These words have very similar meaning in computer science as in common English. One can lock an object, such as locking a gun within a locked cabinet. One can also retain an object, such as keeping a gun under one's mattress or on a shelf in a closet. In both cases, the gun is "kept" or "retained," but, in one case, the gun is secured by locking, and, in the other case, the gun is not locked. The retention or keeping of the gun is not equivalent to securing or locking the gun. Many parents of dead children would probably, in retrospect, wish that "kept" and "locked" meant the same thing, but, unfortunately, they do not.

The Examiner's justification for combining Biliris and Birrell, beginning on line 10 of page 11 of the Examiner's Answer, again makes no sense. The Examiner states, "The Examiner meant to state that one of ordinary skill in the art at the time of the invention would be motivated to use a plurality of buffers, with associated start and end addresses in memory, so that the processor can issue seek commands more efficiently, by skipping to the beginning of one of many buffers, which are separated by a defined granularity, in a circular queue." The Examiner appears to not understand that CODECs must read large amounts of data from consecutive memory addresses in memory that store consecutive data bytes read from a disk. Birrell uses random-access-memory buffers for this purpose. A processor can generally access any byte in a random access memory, and can arrange for a disk-drive controller to do so as well. A processor does not issue "seek" commands to memory. A processor issues "seek" commands to a mass storage device. Seek commands have nothing whatsoever to do with random-memory access, but instead direct a read/write head to a particular track or cylinder of a disk. Moreover, a circular buffer requires a certain amount of overhead. Why would anyone incur this overhead, if one can simply read and write from various locations in random access memory? The Examiner appears to be trying to invent a justification for combining Birrell and Biliris without understanding either of the two references or understanding computer systems and computer science. Using small buffers rather than larger buffers would only complicate Birrell's device, increase computational overhead, and would provide absolutely no advantage that Appellants' representative can imagine. Skipping around a circular queue of small buffers would be far less efficient than transferring large blocks of data to linear memory buffers. One generally tries to minimize issuing seek commands when controlling a disk-drive, because seek commands are expensive in time and incurred latencies. Instead, one generally tries to issue as few seek commands as possible, and read as much data in one read operation as possible.

CONCLUSION

The Examiner's new arguments in the Examiner's Answer make no sense. It is Appellants' representative's impression that the Examiner has failed to understand the current application and the cited references, and is not sufficiently familiar with the fundamentals of computer science and computer systems to properly examine the current application. The Examiner has failed to make a *prima facie* case of obviousness which, according to M.P.E.P. §2142, requires that the Examiner find a teaching or suggestion for all claim limitations in the references combined for the rejection, and the new arguments in the Examiner's Answer only more clearly indicate this failure.

Appellants respectfully submit that all statutory requirements are met and that the present application is allowable over all the references of record. Therefore, Appellants respectfully requests that the present application be passed to issue.

Respectfully submitted,
Eric P. Gibbs et al.
OLYMPIC PATENT WORKS PLLC

By [illegible]
Robert W. Bergstrom
Registration No. 39,906

Olympic Patent Works PLLC
P.O. Box 4277
Seattle, WA 98104
206.621.1933 telephone
206.621.5302 fax